Application Note

MMCSD Debug Guidelines

Shiou Mei Huang

ABSTRACT

This application note describes common debug users may experience when interfacing with enhance Multimedia Card/Secure Digital (eMMC/SD), and Secure Digital I/O (SDIO) devices on the DRA7x and Jacinto 7 family of devices. The first few sections describe key configurations to pay attention to when operating with the module; this may include register settings to adjust timing and PU/PD, as well as programming sequences that you should follow. The subsequent sections list out common problems. Each problem is presented with an issue prompt and a list of recommended debug steps.

Table of Contents

Trademarks

Wi-Fi® is a registered trademark of Wi-Fi Alliance.
All trademarks are the property of their respective owners.

1 Introduction

DRA7x MMC Host Controllers are compliant with:

- JEDEC eMMC Electrical Standard v4.5
- SD Host Controller Specification v3.00, SD Physical Layer Specification v3.01
- SDIO Specification v3.00

Jacinto 7 MMC Host Controllers are compliant with:

- JEDEC eMMC Electrical Standard v5.1
- SD Host Controller Specification v4.10, SD Physical Layer Specification v3.01
- SDIO Specification v3.00

There is more than one MMC modules in both DRA7x and Jacinto 7 devices. Each may support a different specification. To understand which module supports which specification, see the device-specific Technical Reference Manual (TRM).

2 MMC Module Selection

DRA7x and Jacinto 7 MMC controllers support the following frequency of operation at 1.8 V I/O voltage level. Those intending to run at the highest performance should use the MMC module that supports the desired speed mode. For any speed limitations, see the latest device-specific errata document.

Table 2-1. DRA7x MMC Controllers Mode Support

	SD	eMMC	SDIO
MMC1	Yes (up to SDR104 mode)	Yes (up to high-speed DDR mode; timings optimized for SD)	Yes (timings optimized for SD)
MMC2	Yes (up to SDR25 mode: timings optimzed for eMMC)	Yes (up to HS200 mode)	Yes (timings optimized for eMMC)
MMC3	Yes (up to SDR50 mode)	Yes (up to high-speed SDR mode; timings optimized for SD/ SDIO)	Yes (up to SDR50 mode)
MMC4	Yes (up to SDR25 mode)	Yes (up to high-speed SDR mode; timings optimzed for SD/ SDIO)	Yes (up to SDR25 mode)

Table 2-2. Jacinto 7 MMC Controllers Mode Support

	SD/SDIO	eMMC
MMC0	No	Yes (up to HS400 on select parts)
MMC1	Yes (up to SDR104 mode)	No
MMC2	Yes (up to SDR104 mode)	No

Once a target device is selected, runtime operating conditions can be modified to match the operation. This can typically be done by modifying the Processor SDK device tree file. Configurable parameters include pinmux, bus-width, speed modes, voltage level, and max operating frequency. Make sure the target device meets the corresponding frequency requirement.

Device Tree File on DRA7x Processor SDK Linux:

```
&mmc2 {
   status = "okay";
   max-frequency = <192000000>;
   bus-width = <8>
   sd-uhs-sdr25;
   sd-uhs-sdr12;
   mmc-hs200-1_8v;
   mmc-ddr-1_8v;
};
```

Device Tree File on J721E Processor SDK Linux:

```
main_sdhci0: sdhci@4f80000 {
    compatible = "ti,j721e-sdhci-8bit";
    reg = <0x0 0x4f80000 0x0 0x1000>, <0x0 0x4f88000 0x0 0x400>;
    interrupts = <GIC_SPI 3 IRQ_TYPE_LEVEL_HIGH>;
    power-domains = <&k3_pds 91 TI_SCI_PD_EXCLUSIVE>;
    clock-names = "clk_xin", "clk_ahb";
    clocks = <&k3_clks 91 1>, <&k3_clks 91 0>;
    assigned-clocks = <&k3_clks 91 1>;
    assigned-clock-parents = <&k3_clks 91 2>;
    bus-width = <8>;
    ti,otap-del-sel = <0x6>; // Use 0x6 for HS200; 0x0 for HS400
    ti,trm-icp = <0x8>;
    ti,strobe-sel = <0x77>;
    dma-coherent;
};
```

3 Parameters To Configure

3.1 I/O Pad Configuration (Timing, Drive Strength)

To ensure proper operation, certain pin multiplexing and I/O delay configurations should be set as required by the device-specific data manual. The I/O delay timing configuration can be set on either DRA7x or Jacinto 7 devices as described below.

3.1.1 DRA7x I/O Pad Timing

Table 3-1. DRA7x I/O Pad Timing

eMMC/SD/SDIO	
No Virtual or Manual IO Timing Mode Required	MMC1 DS (Pad Loopback) and SDR12 (Pad Loopback) Timings
MMC1_VIRTUAL1	MMC1 HS (Internal Loopback and Pad Loopback), SDR12 (Internal Loopback), SDR25 Timings (Internal Loopback and Pad Loopback)
MMC1_VIRTUAL2	SDR50 (Pad Loopback) Timings
MMC1_VIRTUAL5	MMC1 DS (Internal Loopback) Timings
MMC1_VIRTUAL6	MMC1 SDR50 (Internal Loopback) Timings
MMC1_VIRTUAL7	MMC1 DDR50 (Internal Loopback) Timings
MMC1_DDR_MANUAL1	MMC1 DDR50 (Pad Loopback) Timings
MMC1_SDR104_MANUAL1	MMC1 SDR104 Timings
No Virtual or Manual IO Timing Mode Required	MMC2 Standard (Pad Loopback), High Speed (Pad Loopback), and DDR (Pad Loopback) Timings
MMC2_DDR_LB_MANUAL1	MMC2 DDR (Internal Loopback) Timings
MMC2_HS200_MANUAL1	MMC2 HS200 Timings
MMC2_STD_HS_LB_MANUAL1	MMC2 Standard (Internal Loopback), High Speed (Internal Loopback) Timings
MMC3_MANUAL1	MMC3 DS, SDR12, HS, SDR25 Timings, SDR50 Timings
MMC4_MANUAL1	MMC4 SDR12, HS, SDR25 Timings
MMC4_DS_MANUAL1	MMC4 DS Timings

To achieve a working setup on DRA7x devices, the correct MANUAL or VIRTUAL timing modes has to be programmed in the PADCONFIG registers.

Example:

MMC2_HS200_MANUAL1 setting enables HS200 timing. This setting modifies both agnostic delay (A_DELAY) and gnostic delay (G_DELAY), which are used to calculate total coarse and fine delay elements on the I/O lines.

Table 3-2. Manual Functions Mapping for MMC2 With Internal Loopback Clock and for HS200

BALL	BALL NAME	MMC2_DDR_LB_MANUAL1		MMC2_STD_HS_LB_MANUAL1		MMC2_HS200_MANUAL1			MUXMODE
		A_DELAY (ps)	G_DELAY (ps)	A_DELAY (ps)	G_DELAY (ps)	A_DELAY (ps)	G_DELAY (ps)	CFG REGISTER	1
K7	gpmc_a19	49	0	850	0	-	-	CFG_GPMC_A19_IN	mmc2_dat4
K7	gpmc_a19	0	0	0	0	274	0	CFG_GPMC_A19_OEN	mmc2_dat4
K7	gpmc_a19	170	0	0	0	162	0	CFG_GPMC_A19_OUT	mmc2_dat4
M7	gpmc_a20	463	0	1264	0	-	-	CFG_GPMC_A20_IN	mmc2_dat5
M7	gpmc_a20	0	0	0	0	401	0	CFG_GPMC_A20_OEN	mmc2_dat5
M7	gpmc_a20	81	0	0	0	73	0	CFG_GPMC_A20_OUT	mmc2_dat5
J5	gpmc_a21	8	0	786	0	-	-	CFG_GPMC_A21_IN	mmc2_dat6

In the reference device tree file below, pinctrl-3 sets IODELAY_HS200_1_8V I/O delay configuration for 1.8 V HS200 operation.

Configuration Example:

```
&mmc2 {
  status = "okay";
  pinctrl-names = "default", "hs", "ddr_1_8v", "hs200_1_8v";
  pinctrl-0 = <&mmc2_pins_default>;
  pinctrl-1 = <&mmc2_pins_hs>;
  pinctrl-2 = <&mmc2_pins_ddr_1_8v &mmc2_iodelay_ddr_1_8v_conf>;
  pinctrl-3 = <&mmc2_pins_hs200_1_8v &mmc2_iodelay_hs200_1_8v_conf>;
};
```

Details regarding how to set the I/O delay values in CTRL_CORE_PAD register can be found in the *IOSETS* chapter of the *DRA75x, DRA74x SoC for Automotive Infotainment Silicon Revision 2.0, 1.1 Technical Reference Manual*. The section contains guidance on both Virtual and Manual timing delay modes, including how to translate the given agnostic and gnostic delay values into coarse and fine delay elements to program into IODELAYCONFIG registers.

To select a Virtual IO Timing Mode, the MODESELECT field of each associated CTRL_CORE_PAD register must be set to 0x1, and DELAYMODE field must be set to match the *Virtual Function Mapping* tables in the Data Manual. Selection of all Virtual IO Timings Modes should be done as part of the IO Delay Recalibration Sequence.

Bits	Field Name	Description	Type	Reset
8	GPMC_A19_MODESELECT	Selects between the Default IO Timing Mode and a Virtual or Manual IO Timing Mode. Refer to the device Data Manual for definition of the required settings for a given mode of operation. When this bit is 0b1, a Virtual IO Timing Mode can be selected via the DELAYMODE field of this register, as described in Section 18.4.6.1.5, Virtual IO Timing Modes. Manual IO Timing Modes are selected via the procedure described in Section 18.4.6.1.6, Manual IO Timing Modes. 0x0: Default IO Timing Mode is used 0x1: A Virtual or Manual IO Timing Mode is used	RW	0x0
7:4	GPMC_A19_DELAYMODE	This bit field selects the Virtual Timing Mode used when the MODESELECT bit is set to 0b1. See Section 18.4.6.1.5, Virtual IO Timing Modes for details.	RW	0x0
3:0	GPMC_A19_MUXMODE	0x0: gpmc_a19 0x1: mmc2_dat4 0x2: gpmc_a13 0x4: vin4a_d12 0x6: vin3b_d0 0xE: gpio2_9 0xF: Driver off	RW	0xF

Figure 3-1. Register Call Summary for Register CTRL_CORE_PAD_GPMC_A19

To select a Manual IO Timing Mode, the MODESELECT field of each associated CTRL_CORE_PAD register must be set to 0x1 and the associated IODELAYCONFIG CFG_x_IN, CFG_x_OEN, and CFG_x_OUT registers should be set via the sequence described in the device-specific technical reference manual.

Address Offset	0x0000 01C8		
Physical Address	0x4844 A1C8	**Instance**	IODELAYCONFIG
Description	Delay Select Value in binary coded form for cfg_gpmc_a23_in interface		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED								SIGNATURE								RESERVED	LOCK_BIT	BINARY_DELAY													

Bits	Field Name	Description	Type	Reset
31:18	RESERVED		R	0x0
17:12	SIGNATURE	Write to this register will succeed only if data on these bits carries a signature of 6'h29 (6'b101001)	RW	0x0
11	RESERVED		R	0x0
10	LOCK_BIT	When '1', prevents HW update to this MMR. When '0', allows HW update of this MMR.	RW	0x0
9:0	BINARY_DELAY	Delay Select Value in binary coded form. Indicates the number of coarse (9:5) and fine (4:0) delay elements to be used on the delay line connected to this pad.	RW	0x0

Figure 3-2. CFG_GPMC_A23_IN

3.1.2 DRA7x I/O Pad Drive Strength

While operating in MMC mode, I/O driver strength can be modified by changing the MMCHS_AC12[21:20] DS_SEL register bit field of the corresponding MMCSD module.

Bits	Field Name	Description	Type	Reset
21:20	DS_SEL	Driver Strength Select Host Controller output driver in 1.8V signaling is selected by this bit. In 3.3V signaling, this field is not effective. This field can be set depending on Driver Type A, C and D support bits (DTA, DTC and DTD respectively) in the MMCHS_CAPA2 register. This bit depends on setting of Preset Value Enable. If Preset Value Enable = 0, this field is set by Host Driver. If Preset Value Enable = 1, this field is automatically set by a value specified in the one of Preset Value registers, see, Table 25-22. 0x0: Driver Type B is selected (Default) 0x1: Driver Type A is selected 0x3: Driver Type D is selected 0x2: Driver Type C is selected	RW	0x0

Figure 3-3. MMCHS_AC12

On MMC1, when the I/Os are not configured for MMC operation, CTRL_CORE_CONTROL_HYST_1[30:29] SDCARD_IC register field should be used to control the drive strength.

30:29	SDCARD_IC	Drive strength control for MMC1 pads In 3.3V signaling mode: 0x0: 50 Ohms Drive Strength 0x1: 33 Ohms Drive Strength 0x2: 66 Ohms Drive Strength 0x3: Reserved In 1.8V signaling mode: 0x0: 44 Ohms Drive Strength 0x1: 33 Ohms Drive Strength 0x2: 58 Ohms Drive Strength 0x3: 100 Ohms Drive Strength	RW	0x0

Figure 3-4. CTRL_CORE_CONTROL_HYST_1

3.1.3 Jacinto7 I/O Pad Timing

To meet proper timing, input and output delay on Jacinto 7 devices can be configured by changing the ITAPDLYSEL and OTAPDLYSEL register fields in MMCSDx_SS_PHY_CTRL_4_REG. At the publication of this document, the recommended values for J721E MMC0 are shown in Table 3-3. For the latest recommendation for each respective Jacinto 7 family device, see the device-specific data manual.

Table 3-3. MMC0 DLL Delay Mapping for all Timing Modes

Register Name		MMCSD0_SS_PHY_CTRL_4_REG					MMCSD0_SS_PHY_CTRL_5_REG		
Bit Field		[31:24]	[20]	[15:12]	[8]	[4:0]	[17:16]	[10:8]	[2:0]
Bit Field Name		STRBSEL	OTAPDLYENA	OTAPDLYSEL	ITAPDLYENA	ITAPDLYSEL	SELDLYTXCLK SELDLYRXCLK	FRQSEL	CLKBUFSEL
Mode	Description	Strobe Delay	Output Delay Enable	Output Delay Value	Input Delay Enable	Input Delay Value	DLL/ DELAY CHAIN SELECT	DLL REF FREQUENCY	DELAY BUFFER DURATION
Legacy SDR	8-bit PHY operating 1.8 V, 25 MHz	0x0	0x0	NA	0x1	0x10	0x1	0x0	0x7
High Speed SDR	8-bit PHY operating 1.8 V, 50 MHz	0x0	0x0	NA	0x1	0xA	0x1	0x0	0x7
High Speed DDR	8-bit PHY operating 1.8 V, 50 MHz	0x0	0x1	0x5	0x1	0x3	0x0	0x4	0x7
HS200	8-bit PHY operating 1.8 V, 50 MHz	0x0	0x1	0x6	0x1	Tuning	0x0	v	0x0

In the device tree file referenced below, *ti,otap-del-sel* and *ti,itap-del-sel* define the OTAPDLYSEL and ITAPDLYSEL values to use for the respective speed modes. HS200 does not have a set ITAPDLYSEL value defined, as a tuning algorithm is used on the input.

Configuration Example:

```
main_sdhci0: mmc@4f80000 {
    compatible = "ti,j721e-sdhci-8bit";
    reg = <0x0 0x4f80000 0x0 0x1000>, <0x0 0x4f88000 0x0 0x400>;
    interrupts = <GIC_SPI 3 IRQ_TYPE_LEVEL_HIGH>;
    power-domains = <&k3_pds 91 TI_SCI_PD_EXCLUSIVE>;
    clock-names = "clk_ahb", "clk_xin";
    clocks = <&k3_clks 91 0>, <&k3_clks 91 1>;
    assigned-clocks = <&k3_clks 91 1>;
    assigned-clock-parents = <&k3_clks 91 2>;
    bus-width = <8>;
    mmc-hs200-1_8v;
    mmc-ddr-1_8v;
    ti,otap-del-sel-ddr52 = <0x5>;
    ti,otap-del-sel-hs200 = <0x6>;
    ti,itap-del-sel-legacy = <0x10>;
    ti,itap-del-sel-mmc-hs = <0xa>;
    ti,itap-del-sel-ddr52 = <0x3>;
    ti,trm-icp = <0x8>;
    dma-coherent;
};
```

3.1.4 Jacinto 7 I/O Pad Drive Strength

The I/O driver strength for MMC0 can be modified by changing the MMCSD0_HOST_CONTROL2[5:4] DRIVER_STRENGTH1 register bit field, where:

- Type B: 50 Ω Nominal
- Type A: 33 Ω Nominal
- Type C: 66 Ω Nominal
- Type D: 100 Ω Nominal

Table 3-4. MMCSD0_HOST_CONTROL2 Register Field Descriptions

Bit	Field	Type	Reset	Description
8	UHS2_INTF_ENABLE	R/W	0h	UHS-II Interface Enable This bit is used to enable UHS-II Interface. Before trying to start UHS-II initialization, this bit shall be set to 1h. Before trying to start SD mode initialization, this bit shall be set to 0h. This bit is used to enable UHS-II IF Detection, Lane Synchronization and In Dormant State in the MMCSD0_PRESENTSTATE register, and to select clock source of either SD mode or UHS-II mode. Host Controller shall not leave unused SD 4-bit Interface lines (CLK, CMD and DAT[3:2]) floating in UHS-II mode by using pull-up or driving to low. When DAT[2] is used as interrupt input in UHS-II mode, DAT[2] of Host Controller is set to input and then DAT[2] of SDIO card is set to output to avoid conflict. 0h: 4-bit SD Interface Enabled 1h: UHS-II Interface Enabled
7	SAMPLING_CLK_ SELECT	R/W	0h	Sampling Clock Select (UHS-I Only) This bit is set by tuning procedure when the MMCSD0_HOST_CONTROL2[6] EXECUTE_TUNING bit is cleared. Writing 1h to this bit is meaningless and ignored. Setting 1h means that tuning is completed successfully and setting 0 means that tuning is failed. Host Controller uses this bit to select sampling clock to receive CMD and DAT. This bit is cleared by writing 0h. Change of this bit is not allowed while the Host Controller is receiving response or a read data block. 0h: Fixed clock is used to sample data 1h: Tuned clock is used to sample data

Table 3-4. MMCSD0_HOST_CONTROL2 Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
6	EXECUTE_TUNING	R/W	0h	Execute Tuning (UHS-I Only) This bit is set to 1h to start tuning procedure and automatically cleared when tuning procedure is completed. The result of tuning is indicated to the MMCSD0_HOST_CONTROL2[7] SAMPLING_CLK_SELECT bit. Tuning procedure is aborted by writing 0h for more detail about tuning procedure. 0h: Not Tuned or Tuning Completed 1h: Execute Tuning
5-4	DRIVER_STRENGTH1	R/W	0h	Driver Strength Select (UHS-I Only) Host Controller output driver in 1.8 V signaling is selected by this bit. In 3.3 V signaling, this field is not effective. This field can be set depends on Driver Type A, C and D support bits in the MMCSD0_CAPABILITIES register. This bit depends on setting of the MMCSD0_HOST_CONTROL2[15] PRESET_VALUE_ENA bit. If MMCSD0_HOST_CONTROL2[15] PRESET_VALUE_ENA = 0h, this field is set by Host Driver. If MMCSD0_HOST_CONTROL2[15] PRESET_VALUE_ENA = 1h, this field is automatically set by a value specified in the one of Preset Value registers (MMCSD0_PRESET_VALUE0 - MMCSD0_PRESET_VALUE10). 0h: Driver Type B is Selected (Default) 1h: Driver Type A is Selected 2h: Driver Type C is Selected 3h: Driver Type D is Selected

On MMC1, the CTRLMMR_SDIO1_CTRL[4:0] DRV_STR register bit field should be used to control the drive strength, where:

DRV_STR Value	Drive Strength
Default	40 Ω
Default + 5d	33 Ω
Default + 5d	50 Ω
Default 10d	66 Ω

Table 3-5. CTRLMMR_SDIO1_CTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	Reserved
4-0	DRV_STR	R/W	X	Selects the SDIO drive strength

3.2 Voltage Supply Level

3.2.1 DRA7x MMC Host Controllers

- Supports eMMC/SD/SDIO operation in 3.3 V and 1.8 V.
  - Voltage switching for eMMC interface at runtime is not supported so voltage supply needs to be at a fixed 3.3 V or 1.8 V.
  - Dynamic switching on SD interface is supported and can be done by following the steps in the *Bus Voltage Selection* chapter in the *DRA75x, DRA74x SoC for Automotive Infotainment Silicon Revision 2.0, 1.1 Technical Reference Manual*.
- HS200 and Ultra-High Speed (UHS) modes such as SDR12, SDR25, DDR50, SDR50, SDR104 can only operate in 1.8 V.

3.2.2 Jacinto 7 MMC Host Controllers

- Supports eMMC operation in 1.8 V only. SD/SDIO operation can operate in both 3.3 V and 1.8 V
  - Voltage switching sequence for SD operation is shown in the Signal Voltage Switch Procedure (for UHS-I) section 12.3.6.5.1.6.1 in the TRM.
- HS200 and Ultra-High Speed (UHS) modes such as SDR12, SDR25, DDR50, SDR50, SDR104 can only operate in 1.8 V.

For SD operation, software will always boot up the card in 3.3 V, and negotiate down the voltage value if the target device supports 1.8 V. Due to this reason, you should utilize separate voltage supplies for cage and SD card I/O voltage.

Configuration Examples:

On DRA7x, proper voltage can be achieved by modifying vmmc and *vmmc_aux* supply definitions in dra7-evm.dts.

```
vmmc supplies voltage to the VDD voltage rail
vmmc_aux supplies voltage to the I/O rail
&mmc2 {
    status = "okay";
    vmmc-supply = <&evm_1v8_vdd_sw>;
    vmmc_aux-supply = <&evm_1v8_io_sw>;
};
```

On Jacinto 7, proper voltage can be achieved by specifying voltage values in the device-specific "common-proc-board.dts" file, (On J7ES, this is k3-j721e-common-proc-board.dts).

```
&main_sdhci1 {
    /* SD/MMC */
    vmmc-supply = <&vdd_mmc1>;
    vqmmc-supply = <&vdd_sd_dv_alt>;
    pinctrl-names = "default";
    pinctrl-0 = <&main_mmc1_pins_default>;
    ti,driver-strength-ohm = <50>;
    disable-wp;
};
```

3.3 Tuning Algorithm for High Frequency Modes

HS400, HS200 and UHS-I SDR104 modes enable eMMC/SD/SDIO devices to operate at max 200 MHz (on DRA7x, 192 MHz). At such high frequency, the data sampling window becomes smaller. Tuning is therefore required to identify an optimal ratio to capture data on the I/O lines.

The tuning algorithm described in device-specific technical reference manual largely follows the sequence specified in the MMC standard. This sequence involves a series of tuning commands sent from the host controller to the target device, each with different sampling clock delay to identify a valid data sampling window. The optimal ratio is then chosen from the valid window.

If you plan to use DRA7x devices, see *MMC DLL Tuning*, the Processor SDK implements this tuning algorithm as part of API *omap_hsmmc_execute_tuning()* in *drivers/mmc/host/omap_hsmmc.c*.

If you plan to use Jacinto 7 devices, see *MMC SW Tuning Algorithm*, the Processor SDK implements this tuning algorithm as part of *sdhci_am654_execute_tuning()* or *sdhci_am654_platform_execute_tuning()* in *drivers/mmc/host/sdhci_am654.c.*

If failures are observed during tuning, make sure the latest software patches are applied, and the algorithm mentioned in the application notes are implemented. If failure persisted, see Section 5.

4 Physical Connections

SD and JEDEC industrial specifications provide guidelines on electrical and mechanical connections of the card and controller for reliable operation. You must follow these guidelines to ensure proper functionality.

TI offers a schematic checklist as well. For more information on the checklist – or to submit your board layout files for review with TI – approach your local Field Application Engineer to understand the review submission process.

A few of the recommended signal integrity best practices include:

- Separate routing layers with GND layers.
- Avoid gaps in ground plane between source and load.
- Wherever a signal goes through a via, have a return GND via very close (within a few mm).
- Avoid cross-talk/coupling. Do not route two signals directly above or below each other.
- Avoid cross-talk/coupling. Route traces with spaces between traces that are 2-3X the trace width.
- Avoid stubs. Have zero stubs on traces.
- Ensure signals are monotonic at inputs
- Ensure impedance match

5 Debug

5.1 Debug Steps Checklist

The steps listed below can either solve the bug or assist you to gather critical information to later debug the failure.

- ☐ Check device errata
  - Apply workaround to items fitting failure description
  - Apply latest patches where applicable
- ☐ Check I/O Pad Configuration are set properly
  - Apply recommended setting for I/O delay timing as indicated in the data manual
- ☐ Check voltage level is stable
  - Probe core and MMC voltage rails; are there any voltage dips?
- ☐ Check for dependency on frequency
  - Does a slower speed mode work?
- ☐ Check for dependency on voltage
  - Does a higher or lower VDD_CORE or MMC I/O voltage work?
- ☐ Check for dependency on temperature
- ☐ Check on additional units or boards
  - Are all systems failing the same way?
  - Does failure follow a specific unit, target device, or board after ABA swap?
- ☐ Enable debug prints and dump out registers
  - Connect to a protocol analyzer if applicable. If not, enable CMDs and RSPs print outs in code
  - Save logs during failing runs
  - Save logs during passing runs (if applicable)
- ☐ Make sure relevant voltage supply is stable
  - Probe close to SoC pin for read; close to target device pins for write
  - Zoom out to capture the whole CMD write, CMD response, DAT sequence, and CLK
  - Zoom in to ensure data is valid and capture the waveform
  - Make sure Signal integrity layout guideline from Section 4 Reviewing The Physical Connections is followed
  - Focus on DAT0 for initial measurements. Expand out to other data lines as applicable
- ☐ Compare passing and failing log and waveform. Save each test in individual files

Table 5-1. Command Format

Description	Start Bit	Transmission Bit	Command Index	Argument	CRC7	End Bit
Bit position	47	46	[45:40]	[39:8]	[7:1]	0
Width (bits)	1	1	6	32	7	0
Value	"0"	"1"	X	X	X	"1"

5.2 Request Debug Assistance

If none of the common bugs and root-causes helped with the issue at hand, contact your local FAE or post to E2E with the answers to items listed on the Debug Steps checklist as well as the following.

5.2.1 Standard Setup Info

- Software revision, patches implemented
  - dts file utilized on the board
  - Operating Condition and Failing Condition (If Different):
    - Frequency/ Speed Mode
    - Bit mode
    - Voltage level
    - Temperature
    - Any unique setup
- SD/MMC memory device used in the system
- Board schematic
- A detailed list of what other peripherals are running on the system during failure occurrence

5.2.2 Failure Info

- Information obtained from Debug Steps Checklist
- Kernel logs with debugs enabled from both passing and failing conditions
  - o If multiple tests were run, save each test individually or have a clear “Test Start” and “Test Stopped” to distinguish each fresh re-test
- MMC Host Controller register dumps from both passing and failing conditions
  - On Jacinto 7 family devices, also provide SS_PHY_CTRL4 and SS_PHY_CTRL5 register info
- A description of what the system is doing during failure
- Waveform captures
  - Show MMC, SD, core voltage supplies
  - Show CLK, CMD, DAT lines during failure
    - Zoom out to capture the whole CMD write, CMD RSP, DAT sequence
    - Zoom in to ensure data is valid
    - Verify signal integrity is decent
    - Focus on DAT0 for the initial measurements. Expand to other data lines if 1-bit mode operation passes, but 4-bit mode operation fails

5.2.3 How to Enable Debug Print Logging on DRA7x

Logging on DRA7x can be done by enabling the following settings:

1. Run ./ti_config_fragments/defconfig_builder.sh.
2. Select SDK_Release_Defconfigs
3. Select ti_sdk_dra7x_release.
4. Enable CONFIG_MMC_DEBUG in menuconfig by running the following configurations:
   a. Make ti_sdk_dra7x_release_defconfig.
   b. Make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- menuconfig
5. Enable VERBOSE_DEBUG in omap_hsmmc.c:
   a. Add #define VERBOSE_DEBUG at the beginning of the file (before #include statements)
6. Change loglevel in the kernel command line:
   a. loglevel=8
   b. log_buf_len=2M

5.2.4 How to Enable Debug Print Logging on Jacinto 7

Logging on Jacinto 7 family devices can be done by enabling the following settings:

1. Run ./ti_config_fragments/defconfig_builder.sh.
2. Select v8 ARM Architecture.
3. Select SDK_Release_Defconfigs.
4. Select ti_sdk_arm64_rt_release.
5. Enable CONFIG_MMC_DEBUG in menuconfig by running the following configurations:
   a. make ti_sdk_arm64_release_defconfig ARCH=arm6
   b. make ARCH=arm64 CROSS_COMPILE=aarch64-none-linux-gnu- menuconfig
   c. Check this should be enabled in menuconfig window ::: Device drivers -> MMC/SD/SDIO support -> [*] MMC host drivers debugging
6. Enable VERBOSE_DEBUG in omap_hsmmc.c:
   a. #define VERBOSE_DEBUG at the beginning of the file (before #include statements)
7. Change loglevel in the kernel command line:
   a. loglevel=8 b. log_buf_len=2M

5.2.5 How to Analyze Waveforms

Waveforms can show exactly what is happening on the bus, and either confirm or rule out signal integrity issues on the system. For the waveforms to be useful in debug, capture clean waveforms with the right amount of information.

Waveforms should have a readable 0 and 1 transitions on the signal lines. If there are too much oscillation, chances are the GND connection are loose or too long. Signals should also be probed with 1.5 GHz, and close to the eMMC/SD/SDIO device when the data is outputted from the SoC, or close to the SoC when the data is outputted from the eMMC/SD/SDIO memory.

Note

Reflections could be seen when probing close to the CLK signal at the SoC pin without any termination resistors.

Figure 5-1. Poor Signal Integrity vs Good Signal Integrity on DATA, CMD, and CLK

The captured waveform should show expected CLK, CMD, CMD RSP, and DAT where relevant. Typical CMD structure is command followed by card response, followed by data block if data is involved.

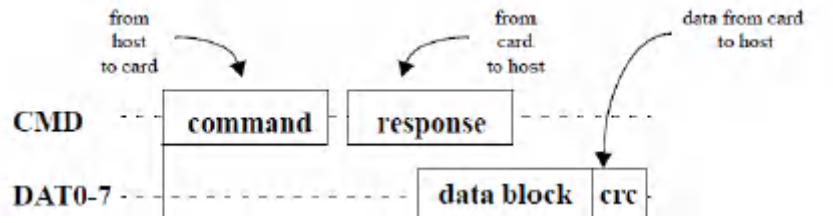

Figure 5-2. CMD Write, CMD Response, and DAT Sequence

In the example below, the system was failing CMD52 (IO_RW_DIRECT) with Command End Bit (CEB) error. Initial customer diagnosis showed Wi-Fi® module was still active at the time of failure, but DRA75x stopped responding.

Upon closer look at the waveform capture, it can be seem DRA75x successfully transmitted CMD52 to the Wi-Fi module, but the Wi-Fi module stopped transmitting data during middle of the transfer; this explained why the CEB error register bit was set. After further deliberation with the firmware provider, it was confirmed the firmware put the Wi-Fi module to sleep during the middle of the transfer and that caused the DRA75x to report error.

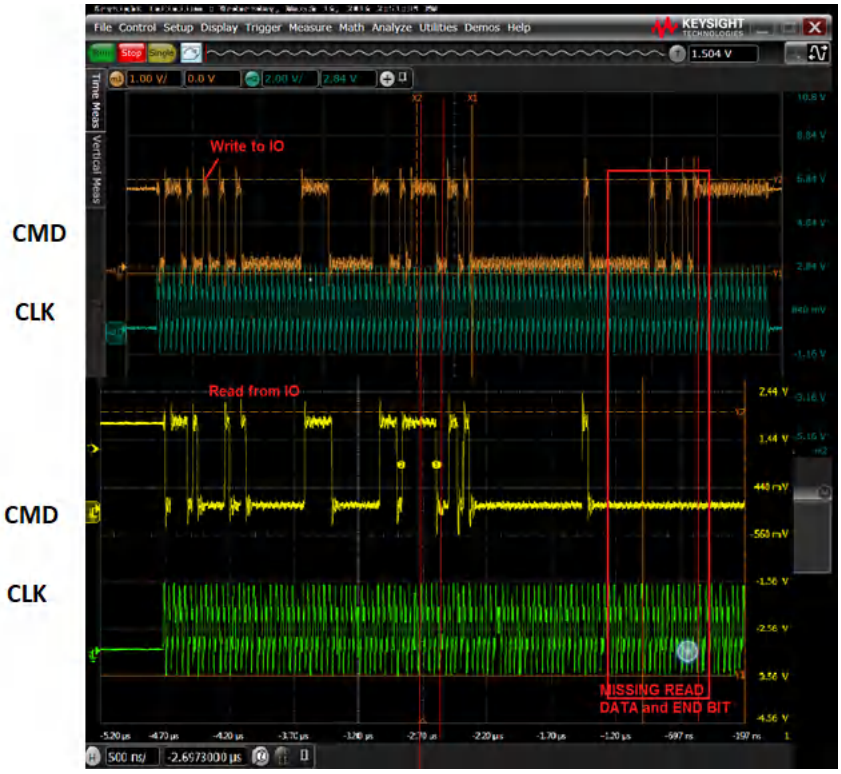

Figure 5-3. CMD Write, CMD Response, and DAT Sequence

5.3 Common Issues

This section lists the common issues experienced while integrating eMMC/SD/SDIO devices with TI DRA7x and Jacinto 7 Host Controllers. Each prompt lists the issue summary, then recommends the debug steps to identify and resolve the issue.

5.3.1 Issue: MMC Did Not Enumerate In Desired Speed Mode

If a system successfully enumerated into the desired speed mode, then the UART log will print out a statement similar to “new HS200 MMC card at address 0001.” If this sentence is not present, then the MMC Host Controller did not enumerate into the desired speed mode. You may see the system enumerated in a slower speed mode, or no successful enumeration at all.

```
[     3.705109] mmc1: new HS200 MMC card at address 0001                  // Card enumerated
successfully at HS200 speed mode
[     3.720436] mmcblk1: mmc1: 0001 MMC08G 7.25 GiB                       // User partition
detected
[     3.730638] mmcblk1boot0: mmc1: 0001 MMC08G partition 1 8.00 MiB      // User partition
detected
[     3.740850] mmcblk1boot1: mmc1: 0001 MMC08G partition 2 8.00 MiB      // User partition
detected
```

Debug steps for this issue will vary based on what enumeration failures happened.

5.3.1.1 System Initialized in Slower Speed Mode

```
[    1.971200] mmc1: SDHCI controller on 4fa0000.sdhci [4fa0000.sdhci] using ADMA 64-bit
[    2.033985] mmc1: new high speed SDHC card at address aaaa
[    2.039948] mmcblk1: mmc1:aaaa SS08G 7.40 GiB
```

If the log does not show any initialization failure and system defaults to a slower speed mode, in this case High Speed, check the device tree entries, the desired speed mode may not have been enabled correctly. The desired speed mode should be listed in device tree and separated with semicolons.

- DRA7x:
  - **SD/SDIO:** sd-uhs-sdr104; sd-uhs-sdr50; sd-uhs-ddr50; sd-uhs-sdr25; sd-uhs-sdr12
  - **eMMC:** mmc-hs200-1_8v; mmc-ddr-1_8v;
- Jacinto 7:
  - **SD/SDIO:** 'sdhci-caps-mask' property corresponds to masking bitfields in MMCSD12_CAPABILITIES. For example, to disable SDR50, DDR50 and SDR104 speed modes, set sdhci-caps-mask = <0xF 0x0>;
  - **eMMC:** mmc-hs200-1_8v, mmc-ddr-1_8v;

5.3.1.2 Speed Mode Initialization Failed

```
[ 2.251614] mmc1: SDHCI controller on fa00000.mmc [fa00000.mmc] using ADMA 64-bit
[ 2.252278] Waiting for root device PARTUUID=9e20d701-02...
[ 2.294370] mmc1: error -110 whilst initialising SD card
[ 2.523388] mmc1: error -110 whilst initialising SD card
[ 3.017091] mmc1: error -110 whilst initialising SD card
[ 33.879361] vdd_mmc1: disabling
```

If the system failed to enter the desired speed mode:

1. Verify the MMC device supports the desired speed mode of operation.
2. Probe the core and MMC voltage rails, confirm the system is operating at a stable voltage and there are no voltage dips on the oscilloscope. Also verify 1.8V voltage level is used for high speed operations.
3. Sanity check I/O delay timing registers are configured properly based on settings recommended by the data manual.
4. Verify the tuning algorithm has been implemented successfully for speed mode greater than 50 MHz, and UHS-I DDR50 in Jacinto 7 family devices,
5. Follow through remaining Debug Step Checklist to determine voltage, temperature, unit dependency, and capture system information.
6. If additional debugs are needed, provide information shown in Request Debug Assistance section of this document. Key information includes logs of both passing and failing scenarios, failure dependency and repeatability, register dumps, as well as waveform captures on the Processors E2E forum.

5.3.1.3 Failed Voltage Switching to 1.8 V

1. Confirm Device and Host both support 1.8 V rail
2. Set MMC CLK frequency to 0.
3. Power Cycle Device.
4. Set Voltage to 1.8 V.
5. Disable PBIAS cell output/power down all PBIAS cell.
6. Follow through the settings of PBIAS register as I mentioned on E2E.
7. Set MMCHS_HCTL[11:9]: SDVS to 0x5 for 1.8 V Support.
8. Set MMCHS_AC12[19]: V1V8_SIGEN to 0x1 for 1.8 V Signaling.
9. Power on the interface by setting MMCHS_HCTL[8] SDBP to 0x1 Power On.
10. You may want to ensure PADEN is set to 0x1 so MMC lines are always active.
11. Send initialize stream and configure the SD card.

5.3.2 Issue: Failed Data Transfer

If the target device failed a transfer, there are multiple possibilities. Isolate where the failure is happening.

1. Determine if failure is a read or write, happening on CLK, CMD, or DAT[7:0] (DAT[3:0]).
   a. What bits are set in the DRA7x PSTATE and STAT registers; Jacinto 7 PRESENTSTATE, NORMAL_INTR_STS, and ERROR_INTR_STS registers?
   b. Does failure always occur at the same location in SW or does it change?
      i. What CMD was transmitted before failure happened?
      ii. Which waveform is last seen on the protocol analyzer or the scope?
   c. Is CLK toggling properly during the failure?
2. Determine if it is a valid failure.
   a. Does the target device still acknowledge all subsequent commands (i.e. CMD13)?
   b. For write failures, are there any data corruptions in the host or target device memory?
   c. For read failures, are there proper or gibberish data on protocol analyzer or scope captures?
   d. Can the system recover and resume transfer after a software reset?
3. Determine if there are environmental dependencies.
   a. Are core and MMC voltage levels stable with no voltage dips?
   b. How many units/ boards show the failure? Is failure occurring only on a specific unit/ board at a specific temperature?
   c. If operating at the highest speed mode, was tuning algorithm run successfully? What ratio was chosen?
4. Follow through with Debug Step Checklist to confirm proper patches and errata workaround have been applied, board signal integrity looks solid, and gather system information
5. If additional debugs are needed, provide information shown in Request Debug Assistance section of this document. Key information includes logs of both passing and failing scenarios, failure dependency and repeatability, register dumps, as well as waveform captures on E2E.

Note

If you have the capability to modify the software, a good way to determine read versus write failure --- during an easily reproducible failure debug --- is as follows:

- Write at the desired frequency, then reduce the frequency and perform a read
- Write at a reduced frequency, then perform a read at the desired frequency

In Linux, standalone read/write tests can be done by executing “dd”:

1. Boot the system.
2. Once at the kernel prompt, ensure that the MMC/SD is enumerated – look for MMC card in the boot logs or dmesg.

```
[    2.856004] mmc1: new HS200 MMC card at address 0001
[    2.864081] VFS: Mounted root (ext4 filesystem) on device 179:2.
[    2.869266] mmcblk1: mmc1:0001 MMC08G 7.25 GiB
```

3. In kernel, do a basic read write test using the following commands.

```
4096+0 records in
4096+0 records out
```

4. If there is no problem in accessing card then there should be no errors and you should see a log like below

6 References

- *DRA75x, DRA74x SoC for Automotive Infotainment Silicon Revision 2.0, 1.1 Technical Reference Manual*
- *DRA829/TDA4VM Technical Reference Manual*

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265